Implementing Pseudonymity

Miranda Mowbray,
miranda.mowbray@hp.com
Technical Expert, HP Labs

Pseudonymity

Allowing individuals to reveal or prove information about themselves to others, without revealing their full identity

- Some functionalities that can't be implemented
- And some that can
- Intuition for the mathematics – but not the detail

Some things technology can't do

- Control the data in your brain
  - Erase my data from your brain
  - Prevent you from using it for marketing
  - Prevent you from communicating it to others
  - Prevent you from combining it with other data you know
- Determine in general what data I can safely reveal to you, without you deducing something I don't want you to know
- Revoke my privacy if I misbehave (where misbehaviour is qualitatively defined), without using a trusted organization

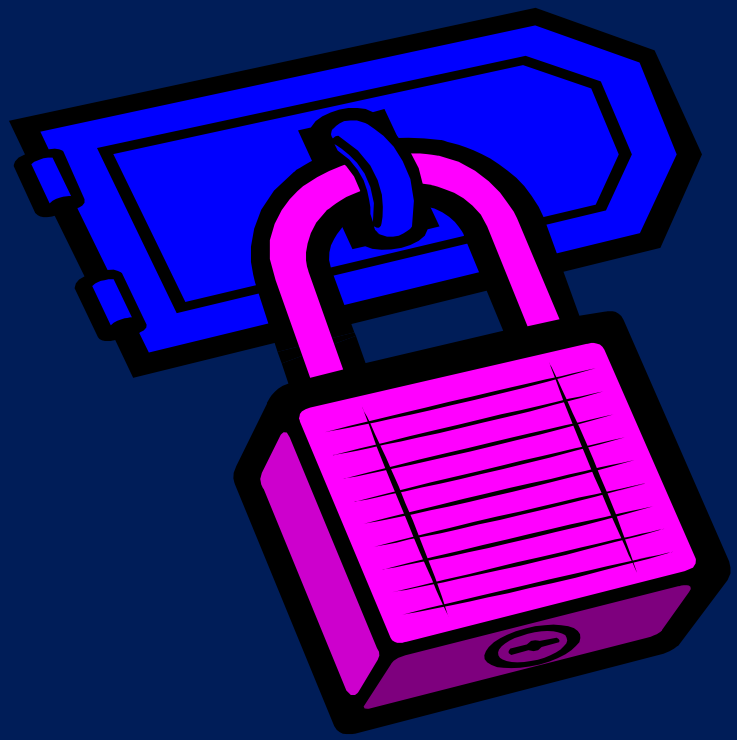
Some things we *can* do

- Prevent pseudonymous ID-theft, without a shared secret
- Unique unlinkable pseudonyms
  - I can only have one pseudonym per organization
  - No-one can link my pseudonyms to my real identity
- Anonymous credentials
  - “This person has a driving licence”
- Release my data only to software checked to conform to particular privacy policies

Main References

- David Chaum (DigiCash), *Security with Identification: Card Computers to make Big Brother Obsolete* Communications of the ACM, vol. 28 no. 10, October 1985 pp. 1030-1044
- Siani Pearson (HP), *Trusted Agents that Enhance User Privacy by Self-Profiling* AAMAS Workshop, special track on privacy,15 July 2002
- E. Brickell (IBM), J. Camenisch (Intel), and L. Chen (HP), *Direct Anonymous Attestation* ACM Conference on Computer and Communications Security (CCS), pp. 132--145, October 2004.

Public Key Cryptography: an analogy



Public key cryptography

- One-way encryption function m -> $f_U(m)$
  - Easy to encrypt using the public key for U
  - Hard to decrypt unless you know the private key
  - eg. $f_U(m)=m^n \pmod r$, n=p.q, p and q good primes
    - Multiplicative: $f_U(m.m')= f_U(m).f_U(m')$
- To send message m privately to U, send $f_U(m)$
- U's signature on c is $f_U^{-1}(c)$
- U can prove he knows the private key by signing a challenge number
  - Actually, U appends a one-time random number and timestamp to the challenge, signs the result, appends random number and timestamp, encrypts the lot in challenger's public key, and sends it to challenger; certificate authority signature for U's public key prevents man-in-the-middle attack

Blind signature

- Basic protocol
  - U picks n, b
    - n of special form such that it's hard for anyone but A to find m with $f_A(m)$ of this form
  - U sends $f_A(b).n$ to A, and requests A's signature
  - A sends $f_A^{-1}(f_A(b).n) = b.f_A^{-1}(n)$ to U
  - U divides this by b to get $f_A^{-1}(n)$
- Anyone can check that $f_A(f_A^{-1}(n))=n$ is of the right form, so that U must have got $f_A^{-1}(n)$ through A
- But the protocol doesn't reveal n to A

Unique unlinkable pseudonyms

- Sketch: Authority A knows U's real identity, but not U's pseudonyms
  - U asks A for a one-time registration key for B
  - U obtains blind signature $f_A^{-1}(n)$ from A
  - Later, U generates a new public/private key pair to use with B
  - U sends $f_A^{-1}(n)$ and public key n_B to B
  - B checks n has the right form, checks with A that n has not already been used
- U can only have one pseudonym with B
- Even if they conspire, A and B can't link n_B to U's real identity (or to any other pseudonym of U)
- Optional timeout for n_B: new pseudonym unlinkable to old

Anonymous credentials

- You use different pseudonyms n_B, n_C with B, C
  - eg. C=driving licence agency, B=car rental agency
- Want to be able to prove to B that C has awarded you a credential, without revealing anything else
- Can do this via a trusted pseudonym-linking organization P
  - Show P credential for n_C, prove to P you know the private keys for both n_B and n_C
  - P vouches that n_B has the credential
- But it can also be done without letting anyone link your pseudonyms

Credential system (David Chaum) - Sketch

- Credential is $f_C^{-1}(n_A)$, a signature by C of your pseudonym with A (A knows your real identity)
- You get the credential using a blind signature protocol
  - C doesn't learn n_A
- To prove you have the credential, have to show some $f_C^{-1}(n)$ and certificate from A that n is of form

  $n = n_A.f_C(s)$ for some s=hash(t), t known to you
- Can obtain certificate for some such n from A
  - A doesn't learn n or s
- To prove to B that you have the credential:
  - compute $f_C^{-1}(n_A).s = f_C^{-1}(n_A.f_C(s)) = f_C^{-1}(n)$
  - send $f_C^{-1}(n)$ and certificate for n to B.

Pseudonymous accountability

- If I misbehave using my pseudonym with B, other organizations might want to know
- Good-behaviour credentials, periodically updated
  - "This person didn't crash any hired cars in 2004"
- Can do more with linkable pseudonyms

  Eg. P: car hire pseudonym -> licence pseudonym if I break the speed limit, P': car hire pseudonym -> my real identity if I run someone over
  - I have to trust P, P' not to abuse their linking powers

Preventing transfer of credentials

- Want to prevent me from transferring my driving licence to unqualified U
  - Can't prevent me from telling U the pseudonym and private key I use with driving licence authority
- Solution 1: (Chen) untransferable single-use-only credentials
- Solution 2: (Bleumer) pseudonym incorporates hashed biometric
- Solution 3: (Camenisch+Lysyanskaya) if U can use any one of my pseudonyms or credentials, U can use them all

Release my data only to software checked to conform to privacy policy

- User data released as f(g(pseudonymous data)), f and g encryption functions
- Only computers with credential that they are one of a set of trusted computing platforms can decrypt f
  - Credential can't be transferred to another computer (derived from secret ID in tamper-resistant hardware)
  - Credentials of rogue computers can be revoked
- On trusted computing platforms only a specific set of software programs, chosen by the user, can decrypt g
  - Eg programs that have been checked to conform to a particular privacy policy
- Data re-encrypted before onward release
- Tracing and auditing of data disclosures
- Still have to trust software developers and environment

Some things we *can* do

- Prevent pseudonymous ID-theft, without a shared secret
- Unique unlinkable pseudonyms
- Anonymous credentials
- Release my data only to software checked to conform to particular privacy policies

invent